Cooperative Convex Optimization in Networked Systems: Augmented Lagrangian Algorithms with Directed Gossip Communication

Dušan Jakovetić, João Xavier, and José M. F. Moura*

Abstract

We study distributed optimization in networked systems, where nodes cooperate to find the optimal quantity of common interest, $x = x^\star$. The objective function of the corresponding optimization problem is the sum of private (known only by a node,) convex, nodes' objectives and each node imposes a private convex constraint on the allowed values of x. We solve this problem for generic connected network topologies with asymmetric random link failures with a novel distributed, decentralized algorithm. We refer to this algorithm as **AL–G** (augmented Lagrangian gossiping,) and to its variants as **AL–MG** (augmented Lagrangian multi neighbor gossiping) and **AL–BG** (augmented Lagrangian broadcast gossiping.) The **AL–G** algorithm is based on the augmented Lagrangian dual function. Dual variables are updated by the standard method of multipliers, at a slow time scale. To update the primal variables, we propose a novel, Gauss-Seidel type, randomized algorithm, at a fast time scale. **AL–G** uses unidirectional gossip communication, only between immediate neighbors in the network and is resilient to random link failures. For networks with reliable communication (i.e., no failures,) the simplified, **AL–BG** (augmented Lagrangian broadcast gossiping) algorithm reduces communication, computation and data storage cost. We prove convergence for all proposed algorithms and demonstrate by simulations the effectiveness on two applications: l_1–regularized logistic regression for classification and cooperative spectrum sensing for cognitive radio networks.

Keywords: Distributed algorithm, convex optimization, augmented Lagrangian, gossip communication

The first and second authors are with the Instituto de Sistemas e Robótica (ISR), Instituto Superior Técnico (IST), 1049-001 Lisboa, Portugal. The first and third authors are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA (e-mail: [djakovetic,jxavier]@isr.ist.utl.pt, moura@ece.cmu.edu, ph: (412)268-6341, fax: (412)268-3890.) This work is partially supported by: the Carnegie Mellon|Portugal Program under a grant from the Fundação de Ciência e Tecnologia (FCT) from Portugal; by FCT grants SIPM PTDC/EEA-ACR/73749/2006 and SFRH/BD/33520/2008 (through the Carnegie Mellon/Portugal Program managed by ICTI); by ISR/IST plurianual funding (POSC program, FEDER); by AFOSR grant # FA95501010291; and by NSF grant # CCF1011903. Dušan Jakovetić holds a fellowship from FCT.

I. INTRODUCTION

Recently, there has been increased interest in large scale networked systems including networks of agents, wireless ad-hoc networks, and wireless sensor networks (WSNs.) Typically, these systems lack a central unit, and the inter-node communication is prone to random failures (e.g., random packet dropouts in WSNs.) In this paper, we consider a generic computational model that captures many applications in networked systems. With this model, nodes cooperate to find the optimal parameter (scalar or vector) of common interest, $x = x^\star$, e.g., the optimal operating point of the network. Each node i has a *private* (known only at node i) cost function of x, e.g., a loss at node i if operating at x. The total cost is the sum over the individual nodes' costs. Also, each node imposes a *private* constraint on the allowed values of x (e.g., allowed operating points at node i.) Applications of this computational model include resource allocation in wireless systems [1], distributed estimation in wireless sensor networks, [2], and distributed, cooperative spectrum sensing in cognitive radio networks, [3], [4].

More formally, nodes cooperatively solve the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x) \\ \text{subject to} & x \in \mathcal{X}_i,\ i = 1, ..., N \end{array} . \tag{1}$$

Here N is the number of nodes in the network, the private cost functions $f_i : \mathbb{R}^m \to \mathbb{R}$ are convex, and each $f_i(\cdot)$ is known locally only by node i. The sets $\mathcal{X}_i$ are *private*, closed, convex constraint sets. We remark that (1) captures the scenario when, in addition to private constraints, there is a public constraint $x \in X$ (where X is a closed, convex set,) just by replacing $\mathcal{X}_i$ with $\mathcal{X}_i \cap X$.

This paper proposes a novel augmented Lagrangian (AL) primal-dual distributed algorithm for solving (1), which handles private costs $f_i(\cdot)$, private constraints $\mathcal{X}_i$, and is resilient to random communication failures. We refer to this algorithm as **AL–G** (augmented Lagrangian gossiping.) We also consider two variants to **AL–G**, namely, the **AL–MG** (augmented Lagrangian multiple neighbor gossiping) and the **AL–BG** (augmented Lagrangian broadcast gossiping.) The **AL–G** and **AL–MG** algorithms use *unidirectional* gossip communication (see, e.g., [5]). For networks with reliable communication (i.e., no failures,) we propose the simplified **AL–BG** algorithm with reduced communication, reduced computation, and lower data storage cost. Our algorithms update the dual variables by the standard method of multipliers, [6], synchronously, at a slow time scale, and update the primal variables with a *novel,* Gauss-Seidel type (see, e.g., [7]) randomized algorithm with asynchronous gossip communication, at a fast time scale. Proof of convergence for the method of multipliers (for the dual variables update) is available in the literature, e.g., [6]. However, our algorithms to update primal variables (referred to as P–AL–G (primal AL gossip), P–AL–MG and P–AL–BG) are novel, a major contribution of this paper is to prove convergence of the

P–AL–G, for private constraints, under very generic network topologies, random link failures, and gossip communication. The proof is then adapted to P–AL–MG and P–AL–BG.

The **AL-G** (and its variants **AL-MG** and **AL-BG**) algorithms are generic tools that fit many applications in networked systems. We provide two simulation examples, namely, l_1–regularized logistic regression for classification and cooperative spectrum sensing for cognitive radio networks. These simulation examples: 1) corroborate convergence of the proposed algorithms; and 2) compare their performance, in terms of communication and computational cost, with the algorithms in [8], [9], [4], [3].

Comparison with existing work. We now identify important dimensions of the communication and computation models that characterize existing references and that help to contrast our paper with the relevant literature. *Optimization algorithms* to solve (1), or problems similar to (1), in a distributed way, are usually either primal-dual distributed algorithms or primal subgradient algorithms. *Optimization constraints* on problem (1) can either be: no constraints ($\mathcal{X}_i = \mathbb{R}^m$); public constraints ($\mathcal{X}_i = \mathcal{X}$); and private constraints $\mathcal{X}_i$. *The underlying communication network* can either be static (i.e., not varying in time,) or dynamic (i.e., varying in time.) A dynamic network can be deterministically or randomly varying. *Link failures* can be symmetric or asymmetric; that is, the random network realizations can be symmetric or, more generally, asymmetric graphs, the latter case being more challenging in general. *The communication protocol* can either be synchronous, or asynchronous, i.e., of gossip type, [5]. We next review the existing work with respect to these four dimensions.

Primal subgradient algorithms. References [10], [8], [11], [9] and [12] develop primal subgradient algorithms, with [10], [8], [11], [9] assuming synchronous communication. References [11] and [9] consider a deterministically varying network, with [11] for the unconstrained problem and [9] for public constraints. References [8] and [10] consider random networks; reference [8] is for public constraints, while [10] assumes private constraints. Both references [10] and [8] essentially handle only *symmetric* link failures, namely, they use local weighted averaging as an intermediate step in the update rule and constrain the corresponding averaging matrix to be *doubly* stochastic. In practice, these translate into requiring symmetric graph realizations, and, consequently, symmetric link failures. Reference [12] presents a primal subgradient algorithm for unconstrained optimization and static network and uses the gossip communication protocol. Finally, reference [13] studies a generalization of problem (1), where the objective function $\sum_{i=1}^N f_i(x)$ is replaced by $g\left(\sum_{i=1}^N f_i(x)\right)$; that is, an outer public, convex function $g(\cdot)$ is introduced. The optimization problem in [13] has public constraints, the communication is synchronous, and the network is deterministically time varying. Reference [13] proposes a distributed algorithm where each node i, at each time step k, updates two quantities: an estimate of the optimal solution $x_i(k)$, and

an estimate of the quantity $(1/N)\sum_{j=1}^{N} f_j(x_i(k))$, by communicating with immediate neighbors only. When the algorithm in [13] is applied to (1), it reduces to the primal subgradient in [9].

Primal-dual algorithms. As far as we are aware, primal-dual algorithms have been studied only for *static networks*. For example, references [4], [3] consider a special case of (1), namely, the Lasso (least-absolute shrinkage and selection operator) type problem. They propose the AD-MoM (alternating direction method of multipliers) type primal-dual algorithms for *static networks*, *synchronous communication*, and *no constraints.* Reference [14] applies AD-MoM to various statistical learning problems, including Lasso, support vector machines, and sparse logistic regression, assuming a parallel network architecture (all nodes communicate with a fusion node,) synchronous communication, and no link failures.

In this paper, rather than subgradient type, we provide and develop a AL primal-dual algorithm for the optimization (1) with private costs and private constraints, random networks, and asynchronous gossip communication. In contrast with existing work on primal-dual methods, for example, [4], [3], our **AL–G** handles *private constraints*, *random networks*, *asymmetric link failures*, and *gossip communication.*[1]

Paper organization. Section II introduces the communication and computational model. Section III presents the **AL–G** algorithm for the networks with link failures. Section IV proves the convergence of the **AL–G** algorithm. Section V studies the variants to **AL–G**, the **AL–MG**, and **AL–BG** algorithms. Section VI provides two simulation examples: 1) l_1–regularized logistic regression for classification; and 2) cooperative spectrum sensing for cognitive radios. Finally, section VII concludes the paper. The Appendix proves convergence of **AL–MG** and **AL–BG**.

II. Problem model

This section explains the communication model (the time slotting, the communication protocol, and the link failures,) and the computation model (assumptions underlying the optimization problem (1).)

Network model: Supergraph. The connectivity of the networked system is described by the bidirectional, connected supergraph $G = (\mathcal{N}, E)$, where $\mathcal{N}$ is the set of nodes (with cardinality $|\mathcal{N}| = N$) and E is the set of bidirectional edges $\{i, j\}$ ($|E| = M$). The supergraph G is simple, i.e., there are no self-edges. Denote by $\Omega_i \subset \mathcal{N}$, the neighborhood set of node i in G, with cardinality $d_i = |\Omega_i|$. The integer d_i is the (supergraph) degree of node i. The supergraph G models and collects all (possibly unreliable) communication channels in the network; actual network realizations during the algorithm run will be *directed* subgraphs of G. We denote the directed edge (arc) that originates in node i and ends in node j either by (i, j) or $i \to j$, as appropriate. The set of all arcs is: $E_d = \{(i, j) : \{i, j\} \in E\}$, where

[1] **AL–G** algorithm uses asynchronous gossip communication, but it is not completely asynchronous algorithm, as it updates the dual variables synchronously, at a slow time scale (as details in Section IV.)

$|E_d| = 2M$. We assume that the supergraph is known, i.e., each node knows a priori with whom it can communicate (over a possibly unreliable link.)

Optimization model. We summarize the assumptions on the cost functions $f_i(\cdot)$ and $f(\cdot)$, $f(x) := \sum_{i=1}^N f_i(x)$, and the constraint sets $\mathcal{X}_i$ in (1):

Assumption 1 We assume the following for the optimization problem (1):

1) The functions $f_i : \mathbb{R}^m \to \mathbb{R}$ are convex and coercive, i.e., $f_i(x) \to \infty$ whenever $\|x\| \to \infty$.
2) The constraint sets $\mathcal{X}_i \subset \mathbb{R}^m$ are closed and convex, and $\mathcal{X} := \cap_{i=1}^N \mathcal{X}_i$ is nonempty.
3) **(Regularity condition)** There exists a point $x_0 \in \text{ri}\,(\mathcal{X}_i)$, for all $i = 1, ..., N$.

Here $\text{ri}\,(\mathcal{S})$ denotes the relative interior of a set $\mathcal{S} \subset \mathbb{R}^m$ (see [15]). We will derive the **AL–G** algorithm to solve (1) by first reformulating it (see ahead eqn. (2),) and then dualizing the reformulated problem (using AL dual.) Assumption 1.3 will play a role to assure strong duality. This will be detailed in subsection III-A. Note that Assumption 1.3 is rather mild, saying only that the intersection of the $\mathcal{X}_i$'s, $i = 1, ..., N$, is "large" enough to contain a point from the relative interior of each of the $\mathcal{X}_i$'s. Denote by $f^\star$ the optimal value and $\mathcal{X}^\star = \left\{x^\star \in \mathcal{X} : \sum_{i=1}^N f_i(x^\star) = f^\star\right\}$ the solution set to (1). Under Assumptions 1, $f^\star$ is finite, and $\mathcal{X}^\star$ is nonempty, compact, and convex, [16]. The model (1) applies also when $\mathcal{X}_i = \mathbb{R}^m$, for i's in a subset of $\{1, ..., N\}$. The functions $f_i(\cdot)$, $f(\cdot)$ need not be differentiable; $f(\cdot)$ satisfies an additional mild assumption detailed in Section IV.

We now reformulate (1) to derive the **AL–G** algorithm. Start by cloning the variable $x \in \mathbb{R}^m$ and attaching a local copy of it, $x_i \in \mathbb{R}^m$, to each node in the network. In addition, introduce the variables $y_{ij} \in \mathbb{R}^m$ and $y_{ji} \in \mathbb{R}^m$, attached to each link $\{i, j\}$ in the supergraph. To keep the reformulated problem equivalent to (1), we introduce coupling constraints $x_i = y_{ij},\ (i, j) \in E_d$ and $y_{ij} = y_{ji},\ \{i, j\} \in E$. The reformulated optimization problem becomes:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & x_i \in \mathcal{X}_i,\ i = 1, ..., N, \\ & x_i = y_{ij},\ (i, j) \in E_d \\ & y_{ij} = y_{ji},\ \{i, j\} \in E. \end{array} \tag{2}$$

The variables x_i and y_{ij} may be interpreted as virtual nodes in the network (see Figure 1.) Physically, the variables x_i, y_{ij}, $j \in \Omega_i$ are maintained by (physical) node i. The virtual link between nodes x_i and y_{ij} is reliable (non-failing,) as both x_i and y_{ij} are physically maintained by node i. On the other hand, the virtual link between y_{ij} and y_{ji} may be unreliable (failing,) as this link corresponds to the physical link between nodes i and j.

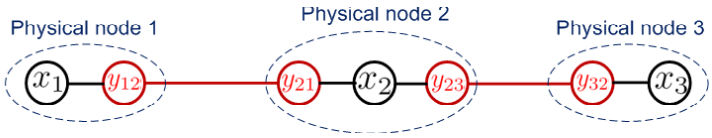

Fig. 1. Illustration of the reformulation (2) for a chain supergraph with $N = 3$ (physical) nodes.

The optimization problems (1) and (2) are equivalent because the supergraph is connected. The optimal value for (2) is equal to the optimal value for (1) and equals $f^\star$; the set of solutions to (2) is $\left\{\{x_i^\star\}, \{y_{ij}^\star\} : x_i^\star = x^\star, \forall i = 1, ..., N,\, y_{ij}^\star = x^\star,\, \forall (i,j) \in E_d, \text{ for some } x^\star \in \mathcal{X}^\star\right\}$.

Time slotting. As we will see in section III, the **AL–G** algorithm (and also its variants **AL–MG** and **AL–BG** in section V) is based on the AL dual of (2). The **AL–G** operates at 2 time scales: the dual variables are updated at a slow time scale, and the primal variables are updated at a fast time scale. Thus, accordingly, the time is slotted with: 1) slow time scale slots $\{t\}$; and 2) fast time scale slots $\{k\}$. Fast time scale slots (for the primal variables update) involve asynchronous communication between the nodes in the network and are detailed in the next paragraph. At the end of each t-slot, there is an idle time interval with no communication, when the dual variables are updated. The dual variables update at each node requires no communication.

Fast time scale slots $\{k\}$ and asynchronous communication model. We now define the fast time scale slots $\{k\}$ for the asynchronous communication and the primal variables update. We assume the standard model for asynchronous communication [5], [17]. Each node (both physical and virtual) has a clock that ticks (independently across nodes) according to a λ-rate Poisson process. Denote the clocks of x_i and y_{ij} by T_i^x and T_{ij}^y, respectively. If T_i^x ticks, a virtual communication from y_{ij}, $\forall j \in \Omega_i$, to x_i, follows. With the **AL–G** algorithm, this will physically correspond to the update of the variable x_i, as we will see later. If the clock T_{ij}^y ticks, then (virtual) node y_{ij} transmits to y_{ji} (physically, node i transmits to node j.) We will see later that, after a (successful) communication $y_{ij} \to y_{ji}$, the update of y_{ji} follows. We also introduce a virtual clock T that ticks whenever one of the clocks T_i^x, T_{ij}^y, ticks; the clock T ticks according to a $(N + 2M)$–rate Poisson process. Denote by τ_k, $k = 1, 2, ...$ the times when the k-th tick of T occurs. The time is slotted and the k-th slot is $[\tau_{k-1}, \tau_k)$, $\tau_0 = 0$, $k = 1, 2, ...$[2]

Random link failures. Motivated by applications in wireless networked systems, we allow that transmissions $y_{ij} \to y_{ji}$ may fail. (Of course, the transmissions through the virtual links $y_{ij} \to x_i$ do not fail.) To formally account for link failures, we define the $N \times N$ random adjacency matrices $A(k)$, $k = 1, 2, ...$; the matrix $A(k)$ defines the set of available physical links at time slot k. We assume that the link failures are

[2]For notation simplicity, at the beginning of each t–slot, we reset τ_0 to zero, and we start counting the k–slots from $k = 1$.

temporally independent, i.e., $\{A(k)\}$ are independent identically distributed (i.i.d.) The entries $A_{ij}(k)$, $(i,j) \in E_d$, are Bernoulli random variables, $A_{ij}(k) \sim \text{Bernoulli}(\pi_{ij})$, $\pi_{ij} = \text{Prob}\,(A_{ij}(k) = 1) > 0$, and $A_{ij}(k) \equiv 0$, for $(i,j) \notin E_d$. We allow $A_{ij}(k)$ and $A_{lm}(k)$ to be correlated.[3] At time slot k, at most one link $(i,j) \in E_d$ is activated for transmission. If it is available at time k, i.e., if $A_{ij}(k) = 1$, then the transmission is successful; if the link (i,j) is unavailable ($A_{ij}(k) = 0$,) then the transmission is unsuccessful. We assume naturally that the Poisson process that governs the ticks of T and the adjacency matrices $A(k)$, $k = 1, 2, ...$ are independent. Introduce the ordering of links $(i,j) \in E_d$, by attaching a distinct number l, $l = 1, ..., 2M$, to each link (i,j); symbolically, we write this as $l \sim (i,j)$. Introduce now the random variables $\zeta(k)$, $k = 1, 2, ...$, defined as follows: 1) $\zeta(k) = i$, if the k-th tick of T comes from T_i^x; 2) $\zeta(k) = N + l$, $l \sim (i,j)$, if the k-th tick of T comes from T_{ij}^y and $A_{ij}(k) = 1$; and 3) $\zeta(k) = 0$, otherwise. It can be shown that $\zeta(k)$, $k = 1, 2, ...$, are i.i.d. The random variables $\zeta(k)$ define the order of events in our communication model. For example, $\zeta(1) = N + l$, $l \sim (i,j)$, means that, at time slot k =1, the virtual node y_{ij} successfully transmitted data to the virtual node y_{ji}. We remark that $\text{Prob}\,(\zeta(k) = s)$ is strictly positive, $\forall s = 0, 1, ..., N + 2M$. This fact will be important when studying the convergence of **AL–G**.

The communication model in this paper, with *static* supergraph and link failures, is standard for networked systems supported by wireless communication and static (non moving) nodes, see, e.g., [18], [19]. The model needs to be modified for scenarios with moving nodes (e.g., mobile robots) where the supergraph itself can be time varying. This is not considered here.

III. **AL–G** ALGORITHM (AUGMENTED LAGRANGIAN GOSSIPING)

This section details the **AL–G** algorithm for solving (1). In subsection III-A, we dualize (2) to form the AL dual of problem (2). Subsection IV-B details the D–AL–G algorithm for the dual variable update, at a slow time scale; subsection IV-C details P–AL–G to update the primal variables, at a fast time scale.

A. Dualization

We form the AL dual of the optimization problem (2) by dualizing all the constraints of the type $x_i = y_{ij}$ and $y_{ij} = y_{ji}$. The dual variable that corresponds to the constraint $x_i = y_{ij}$ will be denoted by $\mu_{(i,j)}$, the dual variable that corresponds to the (different) constraint $x_j = y_{ji}$ will be denoted by $\mu_{(j,i)}$, and the one that corresponds to $y_{ij} = y_{ji}$ is denoted by $\lambda_{\{i,j\}}$. In the algorithm implementation, both nodes i and j will maintain their own copy of the variable $\lambda_{\{i,j\}}$–the variable $\lambda_{(i,j)}$ at node i and the variable $\lambda_{(j,i)}$ at node j. Formally, we use both $\lambda_{(i,j)}$ and $\lambda_{(j,i)}$, and we add the constraint $\lambda_{(i,j)} = \lambda_{(j,i)}$.

[3]With **AL–MG** algorithm, in Section VI, we will additionally require $A_{ij}(k)$ and $A_{lm}(k)$ be independent.

The term after dualizing $y_{ij} = y_{ji}$, equal to $\lambda_{\{i,j\}}^{\top}(y_{ij} - y_{ji})$, becomes: $\lambda_{(i,j)}^{\top} y_{ij} - \lambda_{(j,i)}^{\top} y_{ji}$. The resulting AL dual function $L_a(\cdot)$, the (augmented) Lagrangian $L(\cdot)$, and the AL dual optimization problem are, respectively, given in eqns. (3), (4), and (5).

$$\begin{array}{rlll} L_a\left(\{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right) = & \min & L\left(\{x_i\},\, \{y_{ij}\},\, \{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right) \\ & \text{subject to} & x_i \in \mathcal{X}_i,\, i = 1, ..., N \\ & & y_{ij} \in \mathbb{R}^m,\, (i,j) \in E_d \end{array} \tag{3}$$

$$L\left(\{x_i\}, \{y_{ij}\},\, \{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right) = \sum_{i=1}^{N} f_i(x_i) + \sum_{(i,j)\in E_d} \mu_{(i,j)}^{\top}\,(x_i - y_{ij}) \tag{4}$$

$$+ \sum_{\{i,j\}\in E,\, i<j} \lambda_{(i,j)}^{\top}\, y_{ij} - \lambda_{(j,i)}^{\top}\, y_{ji} + \frac{1}{2}\rho \sum_{(i,j)\in E_d} \|x_i - y_{ij}\|^2 + \frac{1}{2}\rho \sum_{\{i,j\}\in E,\, i<j} \|y_{ij} - y_{ji}\|^2$$

$$\begin{array}{ll} \text{maximize} & L_a\left(\{\lambda_{(i,j)}\},\, \{\mu_{(i,j)}\}\right) \\ \text{subject to} & \lambda_{(i,j)} = \lambda_{(j,i)},\ \{i,j\} \in E\ . \\ & \mu_{(i,j)} \in \mathbb{R}^m,\ (i,j) \in E_d \end{array} \tag{5}$$

In eqn. (4), ρ is a positive parameter. See [6] for some background on AL methods. The terms $\lambda_{(i,j)}^{\top}\, y_{ij} - \lambda_{(j,i)}^{\top}\, y_{ji}$ in the sum $\sum_{\{i,j\}\in E} \lambda_{(i,j)}^{\top}\, y_{ij} - \lambda_{(j,i)}^{\top}\, y_{ji}$ are arranged such that $i < j$, for all $\{i,j\} \in E$. [4]

Denote by $d^\star$ the optimal value of the dual problem (5), the dual of (2). Under Assumption 1, the strong duality between (2) and (5) holds, and $d^\star = f^\star$; moreover, the set of optimal solutions $\mathcal{D}^\star = \left\{\{\lambda_{(i,j)}^\star\},\, \{\mu_{(i,j)}^\star\} :\, L_a\left(\{\lambda_{(i,j)}^\star\},\, \{\mu_{(i,j)}^\star\}\right) = f^\star\right\}$ is nonempty. Denote by $C := \mathcal{X}_1 \times \mathcal{X}_2 \times ... \times \mathcal{X}_N \times (\mathbb{R})^{m(2M)}$ the constraint set in (3), i.e., the constraints in (2) that are not dualized. Let x_0 be a point in $\mathrm{ri}(\mathcal{X}_i)$, $i = 1, ..., N$ (see Assumption 1.3.) Then, a point $(\{x_{i,0}\}, \{y_{ij,0}\}) \in C$, where $x_{i,0} = x_0$, $y_{ij,0} = x_0$, belongs to $\mathrm{ri}(C)$, and it clearly satisfies all equality constraints in the primal problem (2); hence, it is a Slater point, and the above claims on strong duality hold, [15]. We remark that strong duality holds for any choice of $\rho \geq 0$ (but we are interested only in the case $\rho > 0$,) and, moreover, the set of dual solutions $\mathcal{D}^\star$ does not depend on the choice of ρ, provided that $\rho \geq 0$ (see, e.g., [20], p.359.)

[4] For each link $\{i,j\} \in E$, the virtual nodes y_{ij} and y_{ji} (i.e., nodes i and j,) have to agree beforehand (in the network training period) which one takes the $+$ sign and which one takes the $-$ sign in $\lambda_{(i,j)}^{\top}\, y_{ij} - \lambda_{(j,i)}^{\top}\, y_{ji}$. In eqn. (4), for sake of notation simplicity, the distribution of $+$ and $-$ signs at each link $\{i,j\}$ is realized by the order of node numbers, where a distinct number in $\{1, ..., N\}$ is assigned to each node. However, what matters is only to assign $+$ to one node (say i) and $-$ to the other, for each $\{i,j\} \in E$.

B. Solving the dual: D–AL–G (dual augmented Lagrangian gossiping) algorithm

We now explain how to solve the dual problem (5). First, we note that (5) is equivalent to the unconstrained maximization of $L'_a\left(\{\lambda_{\{i,j\}}\},\{\mu_{(i,j)}\}\right) = \min_{x_i\in\mathcal{X}_i, y_{ij}\in\mathbb{R}^m} L'\left(\{x_i\},\{y_{ij}\},\{\lambda_{\{i,j\}}\},\{\mu_{(i,j)}\}\right)$, where the function $L'\left(\{x_i\},\{y_{ij}\},\{\lambda_{\{i,j\}}\},\{\mu_{(i,j)}\}\right)$ is defined by replacing both $\lambda_{(i,j)}$ and $\lambda_{(j,i)}$ in $L(\cdot)$ (eqn. (4)) with $\lambda_{\{i,j\}}$, for all $\{i,j\}\in E$. The standard method of multipliers for the unconstrained maximization of $L'_a(\cdot)$ is given by:

$$\begin{aligned}\lambda_{\{i,j\}}(t+1) &= \lambda_{\{i,j\}}(t) + \rho\ \mathrm{sign}(j-i)\ \left(y^\star_{ij}(t) - y^\star_{ji}(t)\right),\ \{i,j\}\in E \\ \mu_{(i,j)}(t+1) &= \mu_{(i,j)}(t) + \rho\ \left(x^\star_i(t) - y^\star_{ij}(t)\right), (i,j)\in E_d.\end{aligned} \tag{6}$$

$$\begin{aligned}\left(\{x^\star_i(t)\},\{y^\star_{ij}(t)\}\right) \in \text{ arg min } & L'\left(\{x_i\},\{y_{ij}\},\{\lambda_{\{i,j\}}(t)\},\{\mu_{(i,j)}(t)\}\right) \\ \text{subject to } & x_i\in\mathcal{X}_i,\ i=1,...,N \\ & y_{ij}\in\mathbb{R}^m,\ (i,j)\in E_d.\end{aligned} \tag{7}$$

Assigning a copy of $\lambda_{\{i,j\}}$ to both nodes i (the corresponding copy is $\lambda_{(i,j)}$) and j (the corresponding copy is $\lambda_{(j,i)}$), eqn. (6) immediately yields an algorithm to solve (5), given by:

$$\begin{aligned}\lambda_{(i,j)}(t+1) &= \lambda_{(i,j)}(t) + \rho\ \mathrm{sign}(j-i)\ \left(y^\star_{ij}(t) - y^\star_{ji}(t)\right),\ (i,j)\in E_d \\ \mu_{(i,j)}(t+1) &= \mu_{(i,j)}(t) + \rho\ \left(x^\star_i(t) - y^\star_{ij}(t)\right),\ (i,j)\in E_d,\end{aligned} \tag{8}$$

where

$$\begin{aligned}\left(\{x^\star_i(t)\},\{y^\star_{ij}(t)\}\right) \in \text{ arg min } & L\left(\{x_i\},\{y_{ij}\},\{\lambda_{(i,j)}(t)\},\{\mu_{(i,j)}(t)\}\right) \\ \text{subject to } & x_i\in\mathcal{X}_i,\ i=1,...,N \\ & y_{ij}\in\mathbb{R}^m,\ (i,j)\in E_d.\end{aligned} \tag{9}$$

(Note that $\left(\{x^\star_i(t)\},\{y^\star_{ij}(t)\}\right)$ is the same in (7) and (9).) According to eqn. (8), essentially, both nodes i and j maintain their own copy ($\lambda_{(i,j)}$ and $\lambda_{(j,i)}$, respectively) of the same variable, $\lambda_{\{i,j\}}$. It can be shown that, under Assumption 1, any limit point of the sequence $\left(\{x^\star_i(t)\},\{y^\star_{ij}(t)\}\right)$, $t=0,1,...$, is a solution of (2) (see, e.g., [7], Section 3.4); and the corresponding limit point of the sequence $x^\star_i(t)$, $t=0,1,...$, is a solution of (1).

Before updating the dual variables as in (8), the nodes need to solve problem (9), with fixed dual variables, to get $\left(\{x^\star_i(t)\},\{y^\star_{ij}(t)\}\right)$. We will explain in the next subsection (IV-C), how the P–AL–G algorithm solves problem (9) in a distributed, iterative way, at a fast time scale $\{k\}$. We remark that

P–AL–G terminates after a finite number of iterations k, and thus produces an inexact solution of (9). We will see that, after termination of the P–AL–G algorithm, an inexact solution for y_{ji} is available at node i; denote it by $y_{ji}^L(t)$. Denote, respectively, by $x_i^F(t)$ and $y_{ij}^F(t)$, the inexact solutions for x_i and y_{ij} at node i, after termination of P–AL–G. Then, the implementable update of the dual variables is:

$$\begin{aligned} \lambda_{(i,j)}(t+1) &= \lambda_{(i,j)}(t) + \rho \, \mathrm{sign}(j-i) \, \left(y_{ij}^F(t) - y_{ji}^L(t) \right) \\ \mu_{(i,j)}(t+1) &= \mu_{(i,j)}(t) + \rho \, \left(x_i^F(t) - y_{ij}^F(t) \right). \end{aligned} \tag{10}$$

Note that the "inexact" algorithm in (10) differs from (8) in that it does not guarantee that $\lambda_{(i,j)}(t) = \lambda_{(j,i)}(t)$, due to a finite time termination of P–AL–G.

C. Solving for (9)*: P–AL–G algorithm*

Given $\{\lambda_{(i,j)}(t)\}$, $\{\mu_{(i,j)}(t)\}$, we solve the primal problem (9) by a randomized, block-coordinate, iterative algorithm, that we refer to as P–AL–G. To simplify notation, we will write only $\lambda_{(i,j)}$ and $\mu_{(i,j)}$ instead of $\lambda_{(i,j)}(t)$, $\mu_{(i,j)}(t)$. We remark that $\lambda_{(i,j)}(t)$, $\mu_{(i,j)}(t)$ stay fixed while the optimization in eqn. (9) is done (with respect to x_i, y_{ij}.)

The block-coordinate iterative algorithm works as follows: at time slot k, the function in (4) is optimized with respect to a single block-coordinate, either x_i or y_{ij}, while other blocks are fixed. Such an algorithm for solving (9) admits distributed implementation, as we show next. Minimization of the function $L\left(\{x_i\}, \{y_{ij}\}, \{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right)$ with respect to x_i, while the other coordinates x_j and y_{ij} are fixed, is equivalent to the following problem:

$$\begin{array}{ll} \text{minimize} & f_i(x_i) + \left(\overline{\mu}_i - \rho \, \overline{y}_i\right)^\top x_i + \frac{1}{2} \rho \, d_i \|x_i\|^2 \\ \text{subject to} & x_i \in \mathcal{X}_i \end{array}, \tag{11}$$

where $\overline{\mu}_i = \sum_{j \in \Omega_i} \mu_{(i,j)}$ and $\overline{y}_i = \sum_{j \in \Omega_i} y_{ij}$. Thus, in order to update x_i, the node i needs only information from its (virtual) neighbors. Minimization of the function $L\left(\{x_i\}, \{y_{ij}\}, \{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right)$ with respect to y_{ij}, while the other coordinates x_j and y_{lm} are fixed, is equivalent to:

$$\begin{array}{ll} \text{minimize} & \mu_{(i,j)}^\top \, (x_i - y_{ij}) + \lambda_{(i,j)}^\top \mathrm{sign}(j-i) \, (y_{ij} - y_{ji}) + \frac{1}{2}\rho \|x_i - y_{ij}\|^2 + \frac{1}{2}\rho \|y_{ij} - y_{ji}\|^2 \\ \text{subject to} & y_{ij} \in \mathbb{R}^m. \end{array} \tag{12}$$

Thus, in order to update y_{ij}, the corresponding virtual node needs only to communicate information with its neighbors in the network, y_{ji} and x_i. Physical communication is required only with y_{ji} (i.e., with physical node j.) The optimization problem (12) is an unconstrained problem with convex quadratic cost

and admits the closed form solution:

$$y_{ij} = \frac{1}{2} y_{ji} + \frac{1}{2} x_i + \frac{1}{2\rho} \left(\mu_{(i,j)} - \text{sign}(j-i)\, \lambda_{(i,j)} \right). \tag{13}$$

Distributed implementation. We have seen that the block-coordinate updates in eqns. (11) and (13) only require neighborhood information at each node. We next give the distributed implementation of P–AL–G (see Algorithm 1) using the asynchronous communication model defined in section II.

Algorithm 1 Algorithm with gossiping for solving (9) (P–AL–G)

1: **repeat**
2: Wait for the tick of one of the clocks T_j^x, T_{ij}^y.
3: If clock T_{ij}^y ticks, node i transmits to node j the current value of y_{ij}.
If node j successfully receives y_{ij}, it updates the variable y_{ji} according to the equation (13).
4: If clock T_i^x ticks, node i updates the variable x_i by solving (11).
5: **until** a stopping criterion is met.

1) Simplified notation and an abstract model of the P–AL–G: We now simplify the notation and introduce an abstract model for the P–AL–G algorithm, for the purpose of convergence analysis in Section IV. Denote, in a unified way, by z_i, the primal variables x_i and y_{ij}, i.e., $z_i := x_i$, $i = 1, ..., N$, and $z_l := y_{ij}$, $l = N+1, ..., N+2M$, $(i,j) \in E_d$. Then, we can write the function in (4), viewed as a function of the primal variables, simply as $L(z)$, $L : \mathbb{R}^{m(N+2M)} \to \mathbb{R}$. Also, denote in a unified way the constraint sets $C_i := \mathcal{X}_i$, $i = 1, ..., N$, and $C_l := \mathbb{R}^m$, $l = N+1, ..., 2M+N$ (C_l, $l = N+1, ..., N+2M$; these sets correspond to the constraints on y_{ij}, $(i,j) \in E_d$.) Finally, define $C := C_1 \times C_2 \times ... \times C_{N+2M}$. Thus, the optimizations in (11) and (13) are simply minimizations of $L(z)$ with respect to a single (block) coordinate z_l, $l = 1, ..., 2M+N$. Recall the definition of $\zeta(k)$, $k = 1, 2, ...$ in section II. Further, denote $P_i := \text{Prob}\left(\zeta(k) = i\right) > 0$, $i = 0, 1, 2, ..., 2M+N$. Then, it is easy to see that the P–AL–G algorithm can be formulated as in Algorithm 2.

Finally, we summarize the overall primal-dual **AL–G** algorithm in Algorithm 3.

Algorithm 2 AL–G algorithm at node i

1: Set $t = 0$, $\lambda_{(i,j)}(t=0) = 0$, $\mu_{(i,j)}(t=0) = 0$, $j \in \Omega_i$
2: **repeat**
3: Run P–AL–G (cooperatively with the rest of the network) to get $x_i^F(t)$, $y_{ij}^F(t)$ and $y_{ji}^L(t)$, $j \in \Omega_i$
4: Update the dual variables, $\lambda_{(i,j)}(t)$, $\mu_{(i,j)}(t)$, $j \in \Omega_i$, according to eqn. (8).
5: Set $t \leftarrow t+1$
6: **until** a stopping criterion is met.

Remark. With **AL–G**, the updates of the primal variables, on a fast time scale k, are asynchronous and use gossip communication, while the updates of the dual variables, on a slow time scale t, are *synchronous* and require no communication. Physically, this can be realized as follows. Each (physical)

node in the network has a timer, and the timers of different nodes are synchronized. At the beginning of each (slow time scale) t-slot, nodes start the gossip communication phase and cooperatively run the P–AL–G algorithm. After a certain predetermined time elapsed, nodes stop the communication phase and, during an idle communication interval, they update the dual variables. After the idle time elapses, the nodes restart the communication phase at the beginning of the new t-slot.

Choice of ρ. It is known that, under Assumption 1, the method of multipliers (6) converges (i.e., any limit point of the sequence $x_i^\star(t)$, $t = 0, 1, ...$, is a solution of (1)) for any choice of the positive parameter ρ, [21], Theorem 2.1. It converges also if a sequence $\rho_{t+1} \geq \rho_t$ is used, [22], Proposition 4. See [6], 4.2.2, for a discussion on the choice of ρ. The method of multipliers still converges if we use different parameters $\rho = \rho_{(\lambda_{(i,j)},t)}$, $\rho = \rho_{(\mu_{(i,j)},t)}$, for each of the variables $\lambda_{(i,j)}$, $\mu_{(i,j)}$. This corresponds to replacing the quadratic terms $\rho \|x_i - y_{ij}\|^2$ and $\rho \|y_{ij} - y_{ji}\|^2$ in eqn. (4) with $\rho_{(\mu_{(i,j)},t)} \|x_i - y_{ij}\|^2$ and $\rho_{(\mu_{(i,j)},t)} \|y_{ij} - y_{ji}\|^2$, respectively. See reference [23] for details. (We still need $\rho_{(\lambda_{(i,j)},t)} \approx \rho_{(\lambda_{(j,i)},t)}$.) Equation (11) becomes[5]

$$\begin{array}{ll} \text{minimize} & f_i(x_i) + \left(\overline{\mu}_i - \sum_{j\in\Omega_i} \rho_{(\lambda_{(i,j)},t)}\, y_{ij}\right)^\top x_i + \frac{1}{2}\left(\sum_{j\in\Omega_i} \rho_{(\mu_{(i,j)},t)}\right) \|x_i\|^2 \\ \text{subject to} & x_i \in \mathcal{X}_i \end{array} \tag{14}$$

and equation (13) becomes

$$y_{ij} = \frac{\rho_{(\lambda_{(i,j)},t)}}{\rho_{(\lambda_{(i,j)},t)} + \rho_{(\mu_{(i,j)},t)}} (x_i + y_{ji}) + \frac{\mu_{(i,j)} - \text{sign}(j-i)\lambda_{(i,j)}}{\rho_{(\lambda_{(i,j)},t)} + \rho_{(\mu_{(i,j)},t)}}. \tag{15}$$

One possibility for adjusting the parameters $\rho_{(\mu_{(i,j)},t)}$ and $\rho_{(\lambda_{(i,j)},t)}$ in a distributed way is as follows. Each node i adjusts (updates) the parameters $\rho_{(\lambda_{(i,j),t})}$, $\rho_{(\mu_{(i,j),t})}$, $j \in \Omega_i$. We focus on the parameter $\rho_{(\lambda_{(i,j)},t)}$; other parameters are updated similarly. Suppose that the current time is t. Node i has stored in its memory the constraint violation at the previous time $t-1$ that equals $\epsilon_{(\lambda_{(i,j)},t-1)} = \|y_{ij}^F(t-1) - y_{ji}^L(t-1)\|$. Node i calculates the constraint violation at the current time $\epsilon_{(\lambda_{(i,j)},t)} = \|y_{ij}^F(t) - y_{ji}^L(t)\|$. If $\epsilon_{(\lambda_{(i,j)},t)}/\epsilon_{(\lambda_{(i,j)},t-1)} \leq \kappa_{(\lambda_{(i,j)})} < 1$, then the constraint violation is sufficiently decreased, and the parameter $\rho_{(\lambda_{(i,j)},t)}$ remains unchanged, i.e., node i sets $\rho_{(\lambda_{(i,j)},t)} = \rho_{(\lambda_{(i,j)},t-1)}$; otherwise, node i increases the parameter, i.e., it sets $\rho_{(\lambda_{(i,j)},t)} = \sigma_{(\lambda_{(i,j)})}\rho_{(\lambda_{(i,j)},t-1)}$. The constants $\kappa_{(\lambda_{(i,j)})} \in (0,1)$ and $\sigma_{(\lambda_{(i,j)})} > 1$ are local to node i.

[5]Reference [23] proves convergence of the method of multipliers with the positive definite matrix (possibly time-varying) penalty update, see eqn. (1.5) in [23]; the case of different (possibly time-varying) penalties assigned to different constraints is a special case of the matrix penalty, when the matrix is diagonal (possibly time-varying.)

IV. CONVERGENCE ANALYSIS OF THE **AL–G** ALGORITHM

This section provides the convergence of the **AL–G** algorithm. Convergence of the multiplier method for the dual variable updates (on slow time scale $\{t\}$) in eqn. (8) is available in the literature, e.g., [6]. We remark that, in practice, P–AL–G runs for a finite time, producing an inexact solution of (9). This, however, does not violate the convergence of the overall primal-dual **AL–G** scheme, as corroborated by simulations in Section VI. The P–AL–G algorithm for the primal variable update (on the fast time scale $\{k\}$) is novel, and its convergence requires a novel proof. We proceed with the convergence analysis of P–AL–G. First, we state an additional assumption on the function $f(\cdot)$, and we state Theorem 4 on the almost sure convergence of P–AL–G.

Assumptions and statement of the result. Recall the equivalent definition of the P–AL–G and the simplified notation in III-C1. The P–AL–G algorithm solves the following optimization problem:

$$\begin{array}{ll} \text{minimize} & L(z) \\ \text{subject to} & z \in C \end{array} . \tag{16}$$

We will impose an additional, mild assumption on the function $L(z)$, and, consequently, on the function $f(\cdot)$. First, we give the definition of a block-optimal point.

Definition 2 (Block-optimal point) A point $z^\bullet = \left(z_1^\bullet, z_2^\bullet, ..., z_{N+2M}^\bullet\right)$ is block-optimal for the problem (16) if: $z_i^\bullet \in \arg\min_{w_i \in C_i} L\left(z_1^\bullet, z_2^\bullet, ..., z_{i-1}^\bullet, w_i, z_{i+1}^\bullet, ..., z_{N+2M}^\bullet\right)$, $i = 1, ..., N+2M$.

Assumption 3 If a point $z^\bullet$ is a block-optimal point of (16), then it is also a solution of (16).

Remark. Assumption 3 is mild: it is valid if, e.g., $f_i(x) = k_i\|x\|_1 + W_i(x)$, $k_i \geq 0$, where $W_i : \mathbb{R}^m \to \mathbb{R}$ is a continuously differentiable, convex function, and $\|x\|_1 = \sum_{i=1}^N |x_i|$ is the l_1 norm of x, [24].

Define the set of optimal solutions $B = \{z^\star \in C : L(z^\star) = L^\star\}$, where $L^\star = \inf_{z \in C} L(z)$.[6] Further, denote by $\mathrm{dist}(b, A)$ the Euclidean distance of point $b \in \mathbb{R}^m$ to the set $A \subset \mathbb{R}^m$, i.e., $\mathrm{dist}(b, A) = \inf_{a \in A} \|a - b\|_2$, where $\|x\|_2$ is the Euclidean, l_2 norm. We now state the Theorem on almost sure (a.s.) convergence of the P–AL–G algorithm (Theorem 4,) after which we give some auxiliary Lemmas needed to prove Theorem 4.

Theorem 4 Let Assumptions 1 and 3 hold, and consider the optimization problem (16) (with fixed dual variables.) Consider the sequence $\{z(k)\}_{k=0}^\infty$ generated by the algorithm P–AL–G. Then:

[6]Under Assumption 1, the set B is nonempty and compact and $L^\star > -\infty$. This will be shown in Lemma 5. Clearly, $L^\star = L^\star\left(\{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right)$ and $B = B\left(\{\lambda_{(i,j)}\}, \{\mu_{(i,j)}\}\right)$ depend on the dual variables. For simplicity, we write only $L^\star$ and B.

1) $\lim_{k\to\infty} \operatorname{dist}(z(k), B) = 0,$ a.s.

2) $\lim_{k\to\infty} L(z(k)) = L^\star,$ a.s.

Auxiliary Lemmas. Let $i_C : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ be the indicator function of the set C, i.e., $i_C(z) = 0$ if $z \in C$ and $+\infty$ otherwise. It will be useful to define the function $L + i_C : \mathbb{R}^{m(N+2M)} \to \mathbb{R} \cup \{+\infty\}$, $(L + i_C)(z) = L(z) + i_C(z)$. Thus, the optimization problem (16) is equivalent to the unconstrained minimization of $(L + i_C)(\cdot)$. The following Lemma establishes properties of the set of solutions B, the optimal value $L^\star$, and the function $(L + i_C)(\cdot)$.

Lemma 5 Let Assumption 1 hold. The functions $L(z)$ and $(L + i_C)(z)$ are coercive, $L^\star > -\infty$, and the set B is nonempty and compact.

Proof: The function $L(z)$ (given in eqn. (4)) is coercive. To see this, consider an arbitrary sequence $\{z(j)\}_{j=1}^{\infty}$, where $\|z(j)\| \to \infty$ as $j \to \infty$. We must show that $L(z(j)) \to \infty$. Consider two possible cases: 1) there is $i \in \{1, ..., N\}$ such that $\|x_i(j)\| \to \infty$; and 2) there is no $i \in \{1, ..., N\}$ such that $\|x_i(j)\| \to \infty$. For case 1), pick an i such that $\|x_i(j)\| \to \infty$; then $f_i(x_i(j)) \to \infty$, and hence, $L(z(j)) \to \infty$. In case 2), there exists a pair (i, l) such that $\|y_{il}\| \to \infty$; but then, as $x_i(j)$ is bounded, we have that $\|x_i(j) - y_{il}(j)\|^2 \to \infty$, and hence, $L(z(j)) \to \infty$. The function $(L + i_C)(z)$ is coercive because $(L + i_C)(z) \geq L(z)$, $\forall z$, and $L(z)$ is coercive. The function $(L + i_C)(z)$ is a closed[7] (convex) function, because $L(z)$ is clearly a closed function and $i_C(z)$ is a closed function because C is a closed set; hence, $(L + i_C)(z)$ is closed function as a sum of two closed functions.Hence, B is a closed set, as a sublevel set of the closed function $(L + i_C)(z)$. The set B is bounded as a sublevel set of a coercive function $(L + i_C)(z)$. Hence, B is closed and bounded, and thus, compact. We have that $L^\star > -\infty$ (and B is non empty) as $L(z)$ is a continuous, convex, and coercive on $\mathbb{R}^{m(N+2M)}$. ■

Define $U_\epsilon(B) = \{z : \operatorname{dist}(z, B) < \epsilon\}$, and let $V_\epsilon(B)$ be its complement, i.e., $V_\epsilon(B) = \mathbb{R}^m \backslash U_\epsilon(B)$. Further, denote by S and F the initial sublevel sets of L and $L + i_C$, respectively, i.e., $S = \{z : L(z) \leq L(z(0))\}$, and $F = \{z : \ (L + i_C)(z) \leq L(z(0))\} = S \cap C$, where $z(0) \in C$ is a feasible, deterministic, initial point. We remark that, given $z(0)$, any realization of the sequence $\{z(k)\}_{k=0}^{\infty}$ stays inside the set F. This is true because $L(z(k))$ is a nonincreasing sequence by the definition of the algorithm P–AL–G and because any point $z(k)$ is feasible. Define also the set $\Gamma(\epsilon) = F \cap V_\epsilon(B)$. We now remark that, by construction of the P–AL–G algorithm, the sequence of iterates $z(k)$ generated by P–AL–G is a Markov sequence. We are interested in the expected decrease of the function $L(\cdot)$ in one algorithm step, given

[7]A function $q : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is closed if its epigraph $\operatorname{epi}(q) = \{(x, v) : q(x) \leq v\}$ is a closed subset of $\mathbb{R}^{m+1}$.

that the current point is equal to $z(k) = z$:

$$\psi(z) = \mathrm{E}\left[L\left(z(k+1)\right)|z(k) = z\right] - L(z). \tag{17}$$

Denote by $L^i(z)$ the block-optimal value of the function $L(z)$ after minimization with respect to z_i:

$$L^i(z) = \min_{w_i \in C_i} L\left(z_1, z_2, ..., z_{i-1}, w_i, z_{i+1}, ..., z_{N+2M}\right) \tag{18}$$

We have, by the definition of P–AL–G, that (recall the definition of P_i above Algorithm 2:)

$$\psi\left(z\right) \quad = \quad \sum_{i=1}^{N+2M} P_i \left(L^i(z) - L(z)\right). \tag{19}$$

Define $\phi(z) = -\psi(z)$. From eqn. (19), it can be seen that $\phi(z) \geq 0$, for any $z \in C$. We will show that $\phi(z)$ is strictly positive on the set $\Gamma(\epsilon)$ for any positive ϵ.

Lemma 6

$$\inf_{z \in \Gamma(\epsilon)} \phi(z) = a(\epsilon) > 0 \tag{20}$$

We first show that $\Gamma(\epsilon)$ is compact and that L^i is continuous on $\Gamma(\epsilon)$ (latter proof is in the Appendix.)

Lemma 7 (Compactness of $\Gamma(\epsilon)$) The set $\Gamma(\epsilon)$ is compact, for all $\epsilon > 0$.

Proof: We must show that $\Gamma(\epsilon)$ is closed and bounded. It is closed because it is the intersection of the closed sets F and $V_\epsilon(B)$. It is bounded because $\Gamma(\epsilon) \subset F$, and F is bounded. The set F is bounded as a sublevel set of the coercive function $L + i_C$. The set F is closed as a sublevel set of the closed function $L + i_C$. ■

Lemma 8 (Continuity of L^i) The function $L^i : \Gamma(\epsilon) \to \mathbb{R}$ is continuous, $i = 1, ..., N + 2M$.

Proof of Lemma 6: First, we show that $\phi(z) > 0$, for all $z \in \Gamma(\epsilon)$. Suppose not. Then, we have: $L^i(z) = L(z)$, for all i. This means that the point $z \in \Gamma(\epsilon)$ is block-optimal; Then, by Assumption 3, the point z is an optimal solution of (16). This is a contradiction and $\phi(z) > 0$, for all $z \in \Gamma(\epsilon)$. Consider the infimum in eqn. (20). The infimum is over the compact set and the function $\phi(\cdot)$ is continuous (as a scaled sum of continuous functions $L^i(\cdot)$); thus, by the Weierstrass theorem, the infimum is attained for some $z^\bullet \in \Gamma(\epsilon)$ and $\phi(z^\bullet) = a(\epsilon) > 0$. ■

Proof of Theorem 4–1. Recall the expected decrease of the function $L(\cdot)$, $\psi(z)$. We have:

$$\mathrm{E}\left[\psi\left(z(k)\right)\right] = \mathrm{E}\left[\mathrm{E}\left[L\left(z(k+1)\right)|z(k)\right] - L\left(z(k)\right)\right] = \mathrm{E}\left[L\left(z(k+1)\right)\right] - \mathrm{E}\left[L\left(z(k)\right)\right]. \tag{21}$$

On the other hand, we have that $\mathrm{E}\left[\psi(z(k))\right]$ equals:

$$\mathrm{E}\left[\psi\left(z(k)\right)|z(k)\in\Gamma(\epsilon)\right]\,\mathrm{Prob}\left(z(k)\in\Gamma(\epsilon)\right)+\mathrm{E}\left[\psi\left(z(k)\right)|z(k)\notin\Gamma(\epsilon)\right]\,\mathrm{Prob}\left(z(k)\notin\Gamma(\epsilon)\right). \quad (22)$$

Denote by $p_k=\mathrm{Prob}\left(z(k)\in\Gamma(\epsilon)\right)$. Since $\psi\left(z(k)\right)\leq -a(\epsilon)$, for $z(k)\in\Gamma(\epsilon)$, and $\psi\left(z(k)\right)\leq 0$, for any $z(k)$, we have that: $\mathrm{E}\left[\psi\left(z(k)\right)\right]=\mathrm{E}\left[L(z(k+1))\right]-\mathrm{E}\left[L(z(k))\right]\leq -a(\epsilon)\,p_k$; summing up latter inequality for $j=0$ up to $j=k-1$, we get:

$$\mathrm{E}\left[L(z(k))\right]-L(z(0))\leq -a(\epsilon)\sum_{j=0}^{k-1}p_k,\ \forall j\geq 0. \quad (23)$$

The last inequality implies that: $\sum_{k=0}^{\infty}p_k\ \leq\ \frac{1}{a(\epsilon)}\ \left(L\left(z(0)\right)-L^{\star}\right)\ <\ \infty$. Thus, by the first Borel-Cantelli Lemma, $\mathrm{Prob}\left(z(k)\in\Gamma(\epsilon),\ \text{infinitely often}\right)=0$, $\forall\epsilon>0$. Thus, $\mathrm{Prob}\left(\mathcal{A}_{\epsilon}\right)=1$, $\forall\epsilon>0$, where the event $\mathcal{A}_{\epsilon}$ is: $\mathcal{A}_{\epsilon}:=\{\text{the tail of the sequence } z(k) \text{ belongs to } U_{\epsilon}(B)\}$. Consider the event $\mathcal{A}:=\cap_{s=1}^{\infty}\mathcal{A}_{\epsilon_s}$, where ϵ_s is a decreasing sequence, converging to 0. Then, $\mathrm{Prob}\left(\mathcal{A}\right)=\mathrm{Prob}\left(\cap_{s=1}^{\infty}\mathcal{A}_{\epsilon_s}\right)=\lim_{s\to\infty}\mathrm{Prob}\left(\mathcal{A}_{\epsilon_s}\right)=\lim_{s\to\infty}1=1$. Now, the event $\mathcal{B}:=\{\lim_{k\to\infty}\mathrm{dist}(z(k),B)=0\}$ is equal to $\mathcal{A}$, and thus $\mathrm{Prob}\left(\mathcal{B}\right)=1$.

Expected number of iterations for convergence: Proof of Theorem 4–2. Consider now the sets $\mathcal{U}_{\epsilon}(B)=\{z: L(z)\leq\epsilon+L^{\star}\}$ and $\mathcal{V}_{\epsilon}(B)=\mathbb{R}^m\backslash\mathcal{U}_{\epsilon}(B)$ and define the sets $\mathcal{F}$ and $\mathcal{G}(\epsilon)$ as $\mathcal{F}=C\cap S$ and $\mathcal{G}(\epsilon)=\mathcal{F}\cap\mathcal{V}_{\epsilon}(B)$. Similarly as in Lemmas 8, we can obtain that

$$\inf_{z\in\mathcal{G}(\epsilon)}\phi(z)=b(\epsilon)>0. \quad (24)$$

We remark that, once $z(k)$ enters the set $\mathcal{U}_{\epsilon}(B)$ at $k=K_{\epsilon}$, it never leaves this set, i.e., $z(k)\in\mathcal{U}_{\epsilon}(B)$, for all $k\geq K_{\epsilon}$. Of course, the integer K_{ϵ} is random. In the next Theorem, we provide an upper bound on the expected value of K_{ϵ} (the time slot when $z(k)$ enters the set $\mathcal{U}_{\epsilon}(B)$,) thus giving a stopping criterion (in certain sense) of the algorithm P–AL–G.

Theorem 9 (Expected number of iterations for convergence) Consider the sequence $\{z(k)\}_{k=0}^{\infty}$ generated by the algorithm P–AL–G. Then, we have:

$$\mathrm{E}\left[K_{\epsilon}\right]\leq\frac{L\left(z(0)\right)-L^{\star}}{b(\epsilon)}. \quad (25)$$

Proof: Let us define an auxiliary sequence $\widetilde{z}(k)$ as $\widetilde{z}(k)=z(k)$, if $z(k)\in\mathcal{G}(\epsilon)$, and $z(k)=z^{\star}$, if $z(k)\in\mathcal{U}_{\epsilon}(B)$. Here $z^{\star}$ is a point in B. That is, $\widetilde{z}(k)$ is identical to $z(k)$ all the time while $z(k)$ is outside the set $\mathcal{U}_{\epsilon}(B)$ and $\widetilde{z}(k)$ becomes $z^{\star}$ and remains equal to $z^{\star}$ once $z(k)$ enters $\mathcal{U}_{\epsilon}(B)$. (Remark that $z(k)$ never leaves the set $\mathcal{U}_{\epsilon}(B)$ once it enters it by construction of Algorithm P–AL–G.)

Now, we have that:

$$\psi\left(\widetilde{z}(k)\right) = \begin{cases} \psi(z(k)) \leq -b(\epsilon) & \text{if } z(k) \in \mathcal{G}(\epsilon) \\ 0 & \text{if } z(k) \in \mathcal{U}_\epsilon(B) \end{cases}. \tag{26}$$

Taking the expectation of $\psi\left(z(k)\right)$, $k = 0, ..., t-1$ and summing up these expectations, and letting $t \to \infty$, we get:

$$\mathrm{E}\left[L\left(\widetilde{z}(\infty)\right)\right] - L(z(0)) = \sum_{k=0}^{\infty} \mathrm{E}\left[\psi\left(\widetilde{z}(k+1)\right)\right] - \mathrm{E}\left[\psi\left(\widetilde{z}(k)\right)\right] = \mathrm{E}\sum_{k=0}^{\infty} \psi\left(\widetilde{z}(k)\right) \leq -\mathrm{E}\left[K_\epsilon\right] b(\epsilon)$$

Thus, the claim in equation (25) follows. ■

We now prove Theorem 4–2. By Theorem 10, the expected value of K_ϵ is finite, and thus K_ϵ is finite a.s. This means that for all $\epsilon > 0$, there exists random number K_ϵ (a.s. finite), such that $\widetilde{z}(k) = z^\star$, for all $k \geq K_\epsilon$, i.e., such that $z(k) \in \mathcal{U}_\epsilon(B)$ for all $k \geq K_\epsilon$. The last statement is equivalent to Theorem 4–2.

V. Variants to **AL–G**: **AL–MG** (augmented Lagrangian multi neighbor gossiping) and **AL–BG** (augmented Lagrangian broadcast gossiping) algorithms

This section introduces two variants to the **AL–G** algorithm, the **AL–MG** (augmented Lagrangian multi neighbor gossiping) and the **AL–BG** (augmented Lagrangian broadcast gossiping). Relying on the previous description and analysis of the **AL–G** algorithm, this section explains specificities of the **AL–MG** and **AL–BG** algorithms. Subsection V-A details the **AL–MG**, and subsection V-B details the **AL–BG** algorithm. Proofs of the convergence for P–AL–MG and P–AL–BG are in the Appendix.

A. *AL–MG algorithm*

The **AL–MG** algorithm is a variation of the **AL–G** algorithm. The algorithms **AL–G** and **AL–MG** are based on the same reformulation of (1) (eqn.(2)), and they have the same dual variable update (that is, D–AL–G and D–AL–MG are the same.) We proceed by detailing the difference between P–AL–MG and P–AL–G to solve (9) (with fixed dual variables.) With the algorithm P–AL–MG, each node has two independent Poisson clocks, T_i^x and T_i^y. Update followed by a tick of T_i^x is the same as with P–AL–G (see Algorithm 1, step 4.) If T_i^y ticks, then node i transmits *simultaneously* the variables y_{ij}, $j \in \Omega_i$, to all its neighbors (y_{i,j_1} is transmitted to node j_1, y_{i,j_2} is transmitted to node j_2, etc.) Due to link failures, the neighborhood nodes may or may not receive the transmitted information. Successfull transmissions are followed by updates of y_{ji}'s, according to eqn. (13). Define also the virtual clock T that ticks whenever one of the clocks T_i^x, T_i^y, ticks. Accordingly, we define the k-time slots as $[\tau_{k-1}, \tau_k)$, $k = 1, 2...$, $\tau_0 = 0$, and τ_k is the time of the k-th tick of T. Overall **AL–MG** algorithm is the same

as **AL–G** (see Algorithm 3,) except that, instead of P–AL–G, nodes run P–AL–MG algorithms at each t. We prove convergence of the P–AL–MG in the Appendix; for convergence of the overall **AL–MG** algorithm, see discussion at the beginning of section V.

*B. **AL–BG** algorithm: An algorithm for static networks*

We now present a simplified algorithm for the networks with reliable transmissions. This algorithm is based on the reformulation of (1) that eliminates the variables y_{ij}'s. That is, we start with the following equivalent formulation of (1):

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & x_i \in \mathcal{X}_i,\ i=1,...,N, \\ & x_i = x_j,\ \{i,j\} \in E \end{array} \tag{27}$$

We remark that (27) is equivalent to (1) because the supergraph is connected. After dualizing the constraints $x_i = x_j$, $(i,j) \in E$, the AL dual function $L_a(\cdot)$ and the Lagrangian $L(\cdot)$ become:

$$\begin{array}{rlll} L_a\left(\{\lambda_{\{i,j\}}\}\right) = & \min & L\left(\{x_i\},\, \{\lambda_{\{i,j\}}\}\right) \\ & \text{subject to} & x_i \in \mathcal{X}_i,\, i=1,...,N \end{array}$$

$$L\left(\{x_i\},\, \{\lambda_{\{i,j\}}\}\right) = \sum_{i=1}^{N} f_i(x_i) + \sum_{\{i,j\}\in E,\, i<j} \lambda_{\{i,j\}}^{\top}\,(x_i - x_j) + \frac{1}{2}\rho \sum_{\{i,j\}\in E,\, i<j} \|x_i - x_j\|^2. \tag{28}$$

In the sums $\sum_{\{i,j\}\in E} \lambda_{\{i,j\}}^{\top}\,(x_i - x_j)$ and $\sum_{\{i,j\}\in E} \|x_i - x_j\|^2$, the terms $\lambda_{\{i,j\}}^{\top}\,(x_i - x_j)$ and $\|x_i - x_j\|^2$ are included once. (The summation is over the undirected edges $\{i,j\}$.) Also, terms $\lambda_{\{i,j\}}^{\top}\,(x_i - x_j)$ in the sum $\sum_{\{i,j\}\in E} \lambda_{\{i,j\}}^{\top}\,(x_i - x_j)$ are arranged such that $i < j$, for all $\{i,j\} \in E$. The resulting dual optimization problem is the unconstrained maximization of $L_a(\lambda_{\{i,j\}})$.

Solving the dual: D–AL–BG algorithm. We solve the dual (28) by the method of multipliers, which can be shown to have the following form:

$$\lambda_{\{i,j\}}(t+1) = \lambda_{\{i,j\}}(t) + \rho\,\mathrm{sign}(j-i)\,\left(x_i^{\star}(t) - x_j^{\star}(t)\right) \tag{29}$$

$$x^{\star}(t) = (x_1^{\star}(t), x_2^{*}(t), ..., x_N^{*}(t)) \in \begin{array}{lll} \text{arg min} & L\left(\{x_i\},\, \{\lambda_{\{i,j\}}(t)\}\right) \\ \text{subject to} & x_i \in \mathcal{X}_i,\, i=1,...,N \end{array}. \tag{30}$$

We will explain in the next paragraph how the P–AL–BG algorithm solves (30) in a distributed, iterative way. With **AL–BG**, each node needs to maintain only one m-dimensional dual variable: $\overline{\lambda}_i := \sum_{j\in\Omega_i} \mathrm{sign}(j-i)\,\lambda_{\{i,j\}}$. Also, define $\overline{x}_i := \sum_{j\in\Omega_i} x_j$. The P–AL–G algorithm terminates after a finite

number of inner iterations k, producing an inexact solution. Denote by x_i^F (resp. x_j^F) the inexact solution of x_i (resp. x_j, $j \in \Omega_i$), available at node i, after termination of P–AL–BG. We will see that $x_i^F = x_i^L$, $\forall i$; accordingly, after termination of P–AL–BG, node i has available $\overline{x}_i^F := \sum_{j\in\Omega_i} x_j^F$. Summing up equations (29) for $\lambda_{\{i,j\}}$, $j \in \Omega_i$, and taking into account the finite time termination of the P–AL–BG, we arrive at the following dual variable update at node i:

$$\overline{\lambda}_i(t+1) = \overline{\lambda}_i(t) + \rho \left(d_i \, x_i^F(t) - \overline{x}_i^F(t) \right), \; i = 1, ..., N. \tag{31}$$

Solving for (30)**: P–AL–BG algorithm.** We solve the problem (30) by a randomized, block-coordinate P–AL–BG algorithm. After straightforward calculations, it can be shown that minimization of the function in (28) with respect to x_i (while other coordinates are fixed) is equivalent to the following minimization:

$$\begin{array}{ll} \text{minimize} & f_i(x_i) + \left(\overline{\lambda}_i - \rho \, \overline{x}_i\right)^\top x_i + \frac{1}{2} \, \rho \, d_i \|x_i\|^2 \\ \text{subject to} & x_i \in \mathcal{X}_i \end{array} \tag{32}$$

Similarly as with **AL–G**, we assume that the clock ticks at all nodes are governed by independent Poisson process T_i's. P–AL–BG is as follows. Whenever clock T_i ticks, node i updates x_i via eqn. (32) and broadcasts the updated x_i to all the neighbors in the network. Discrete random iterations $\{k\}$ of the P–AL–BG algorithm are defined as ticks of the virtual clock T that ticks whenever one of T_i ticks. The P–AL–BG algorithm produces x_i^F and $\overline{x}_i^F$ at node i. Overall primal-dual **AL–BG** algorithm is similar to the **AL–G** algorithm (see Algorithm 3), except that, at each t, nodes cooperatively run the P–AL–BG algorithm, instead of P–AL–G algorithm. We prove convergence of P–AL–BG in the Appendix; for convergence of the overall primal-dual **AL–BG** scheme, see discussion at the beginning of Section V.

VI. Simulation examples

In this section, we consider two simulation examples, namely, l_1–regularized logistic regression for classification (subsection VI-A,) and cooperative spectrum sensing for cognitive radio networks (subsection VI-B.) Both examples corroborate the convergence of our algorithms **AL–G**, **AL–MG** on random networks, and **AL–BG** on static networks, and demonstrate tradeoffs that our algorithms show with respect to the existing literature. We compare the convergence speed of our and existing algorithms with respect to: 1) communication cost; and 2) computational cost, while the communication cost is dominant in networked systems supported by wireless communication. **AL–BG** outperforms existing algorithms (in [10], [8], [25], [4][8]) on static networks in terms of communication cost, on both examples; at the same time, it has a larger computational cost. For the l_1-regularized logistic regression example and random

[8]Reference [4] focusses specifically on the Lasso problem; we compare with [4] in subsection VI-B.

networks, **AL–G** and **AL–MG** outperform existing algorithms ([10], [8][9]) in terms of communication cost, while having larger computational cost. For the cooperative spectrum sensing example and random networks, **AL–G** and **AL–MG** converge slower than existing algorithms [10], [8].

A. l_1–regularized logistic regression for classification

We consider distributed learning of a linear discriminant function. In particular, we consider the l_1–regularized logistic regression optimization problem (eqn. (45) in [14]; see Subsections 7.1 and 10.2). We add private constraints and adapt the notation from [14] to fit our exposition.[10] The problem setup is as follows. Each node i, $i = 1, ..., N$, has N_d data samples, $\{a_{ij}, b_{ij}\}_{j=1}^{N_d}$, where $a_{ij} \in \mathbb{R}^m$ is a feature vector (data vector,) and $b_{ij} \in \{-1, +1\}$ is the class label of the feature vector a_{ij}. That is, when $b_{ij} = 1$ (respectively, -1,) then the feature vector a_{ij} belongs to the class "1" (respectively, "-1".) The goal is to learn the weight vector $w \in \mathbb{R}^m$, and the offset $v \in \mathbb{R}$, based on the available samples at all nodes, $\{a_{ij}, b_{ij}\}_{j=1}^{N_d}$, $i = 1, ..., N$, so that w is sparse, and the equality: $\text{sign}\left(a_{ij}^\top w + v\right) = b_{ij}$, $i = 1, ..., N$, $j = 1, ..., N_d$, holds for the maximal possible number of data samples $\{a_{ij}, b_{ij}\}_{j=1}^{N_d}$, $i = 1, ..., N$. One approach to choose w and v is via l_1–regularized logistic regression; that is, choose $w^\star$ and $v^\star$ that solve the following optimization problem, [14]:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} \sum_{j=1}^{N_d} \log\left(1 + \exp\left(-b_{ij}(a_{ij}^\top w + v)\right)\right) + \lambda \|w\|_1 \\ \text{subject to} & w^\top w \leq k_i, \ i = 1, ..., N \\ & |v| \leq k_i', \ i = 1, ..., N \end{array} . \tag{33}$$

The parameter $\lambda > 0$ enforces the sparsity in w, [26]. The private constraints on w and v at node i (k_i's and k_i''s are positive) represent the prior knowledge available at node i (see [27], Chapter 7.) Problem (33) clearly fits our generic framework in (1) and has a vector optimization variable, a non smooth objective function, and quadratic private constraints. Alternatives to (33) to learn w and v include support vector machines and boosting, [26], [14].

Simulation setup. We consider a supergraph with $N = 20$ nodes and $|E| = 37$ undirected edges (74 arcs). Nodes are uniformly distributed on a unit square and pairs of nodes with distance smaller than a radius r are connected by an edge. For networks with link failures, the link failures of different arcs at the same time slot are independent and the failures of the same arc at different time slots are independent also. Link failure probabilities π_{ij} are generated as follows: $\pi_{ij} = k\,\frac{\delta_{ij}^2}{r^2}$, $\delta_{ij} < r$, where

[9]Only references [10], [8] consider random networks.

[10]Note that [14] studies only the parallel network architecture, with a fusion center, and it does not propose an algorithm to solve the l_1–regularized logistic regression problem on generic networks, the case that we address here.

$k = 0.5$. Each node has $N_d = 5$ data samples. Each feature vector $a_{ij} \in \mathbb{R}^m$, $m = 20$, and the "true" vector $w_{\rm true}$ have approximately 60% zero entries. Nonzero entries of a_{ij} and $w_{\rm true}$, and the offset $v_{\rm true}$ are generated independently, from the standard normal distribution. Class labels b_{ij} are generated by: $b_{ij} = \text{sign}\left(a_{ij}^\top w_{\rm true} + v_{\rm true} + \epsilon_{ij}\right)$, where ϵ_{ij} comes from the normal distribution with zero mean and variance 0.1. The penalty parameter λ is set to be $0.5 \cdot \lambda_{\max}$, where $\lambda_{\max}$ is the maximal value of λ above which the solution to (33) is $w^\star = 0$ (see ([14], subsection 10.2) how to find $\lambda_{\max}$.) We set k_i and k_i' as follows. We solve the unconstrained version of (33) via the centralized subgradient algorithm; we denote the corresponding solution by $w^\bullet$ and $v^\bullet$. We set $k_i = (1 + r_i) \cdot \|w^\bullet\|^2$, $k_i' = (1 + r_i') \cdot |v^\bullet|$, where r_i and r_i' are drawn from the uniform distribution on $[0, 1]$. Thus, the solution to problem (33) is in the interior of the constraint set. (Similar numerical results to the ones presented are obtained when the solution is at the boundary.) To update x_i with P–AL–G and P–AL–MG (eqn. (11)), we solve (11) via the projected subgradient algorithm.

Algorithms that we compare with. In the first set of experiments, we consider **AL–BG** for (static) networks; in the second set of experiments, we test **AL–G** and **AL–MG** on networks with link failures. We compare our algorithms with the ones proposed in [10], [11], [9], [8][11] and in [25]. References [10], [11], [9], [8] propose a primal projected subgradient algorithm, here refer to as PS (Primal Subgradient.) PS, as an intermediate step, computes weighted average of the optimal point estimates across node i's neighborhood. Averaging weights have not been recommended in [10], [11], [9], [8]; we use the standard time-varying Metropolis weights, see [28], eqn. (11). Reference [25] proposes an incremental primal subgradient algorithm, here referred to as MCS (Markov chain subgradient.) With MCS, the order of incremental updates is guided by a Markov chain, [25].[12] We simulate MCS and PS with fixed subgradient step size rather than the diminishing step size, as the former yields faster convergence.

We compare the algorithms based on two criteria. The first is the amount of inter-neighbor communication that the algorithms require to meet a certain accuracy. We count the total number of radio transmissions (counting both successful and unsuccessful transmissions.) The second is the total number of floating point operations (at all nodes.) In networked systems supported by wireless communication (e.g., WSNs,) the dominant cost (e.g., power consumption) is induced by communication. Total number of floating point operations depends on the algorithm implementation, but the results to be presented give a good estimate of the algorithms' computational cost. It may be possible to reduce the computational

[11]We simulate the algorithms in [10], [11], [9], [8] with symmetric link failures.

[12]Convergence for MCS has been proved only with the projection onto a public constraint set, but we simulate it here with the straightforward generalization of the projection onto private constraint sets; MCS showed convergence for our example in the private constraints case also.

cost of **AL–G**, **AL–MG**, and **AL–BG** by a more computationally efficient solutions to problems (11) and (32) than (here adopted) projected subgradient method.

Denote by $f^\star$ the optimal value of (33). We compare the algorithms in terms of the following metric:

$$\mathbf{err}_f = \frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - f^\star\right),$$

where x_i is the estimate of the optimal solution available at node i at a certain time.

With our **AL–G, AL-MG**, and **AL–BG** algorithms, the simulations to be presented use an increasing sequence of AL penalty parameters (see the end of Section IV,) which, after some experimentation, we set to the following values: $\rho_t = t^{A_\rho} + B_\rho$, $t = 0, 1, ...$, with $A_\rho = 1.3$, and $B_\rho = 1$. We also implemented the algorithms with different and increasing ρ's assigned to each dual variable, with the scheme for adjusting ρ's explained at the end of Section IV, with $\kappa_{\lambda_{(i,j)}} = \kappa_{\mu_{(i,j)}} = 0.3$, and $\sigma_{\lambda_{(i,j)}} = \sigma_{\mu_{(i,j)}} = 1.2$. The latter choice also showed convergence of **AL–G, AL-MG**, and **AL–BG**, but the former yielded faster convergence. Our simulation experience shows that the convergence speed of **AL–G, AL-MG**, and **AL–BG** depend on the choice of ρ_t, but the optimal tuning of ρ_t is left for future studies. With PS and MCS, and a fixed step size, the estimates $f(x_i)$ converge only to a neighborhood of $f^\star$. There is a tradeoff between the limiting error $\mathbf{err}_f(\infty)$ and the rate of convergence with respect to the stepsize α: larger α leads to faster convergence and larger $\mathbf{err}_f(\infty)$. We notice by simulation that **AL–G, AL–MG**, and **AL–BG** converge to a plateau neighborhood of $f^\star$; after that, they improve slowly; call the error that corresponds to this plateau $\mathbf{err}_f(ss)$. To make the comparison fair or in favor of PS and MCS, we set α for the PS and MCS algorithms such that the $\mathbf{err}_f(\infty)$ for PS and MCS is equal (or greater) than the $\mathbf{err}(ss)$ attained by **AL–G, AL–MG**, and **AL–BG**.

Results: Static network. Figure 2 (top left) plots $\mathbf{err}_f$ versus the number of ($m = 20$-dimensional vector) transmissions (cumulatively at all nodes.) We can see that **AL–BG** outperforms PS and MCS by one to two orders of magnitude. **AL–BG** needs about $0.3 \cdot 10^5$ transmissions to reduce $\mathbf{err}_f$ below 0.001, while MCS and PS need, respectively, about $4 \cdot 10^5$ and $18 \cdot 10^5$ transmissions for the same precision. With respect to the number of floating point operations (Figure 2, top right,) **AL–BG** needs more operations than MCS and PS; $45 \cdot 10^8$ for **AL–BG** versus $13 \cdot 10^8$ for PS, and $2 \cdot 10^8$ for MCS. Thus, with respect to MCS, **AL–BG** reduces communication at a cost of additional computation. Note that with **AL–BG**, MCS, and PS, due to private constraints, node i's estimate x_i may not be feasible at certain time slots; in this numerical example, **AL–BG**, MCS, and PS all produced feasible solutions at any time slot, at all nodes. A drawback of MCS in certain applications, with respect to PS and **AL–BG**, can be the *delay time* that MCS needs for the "token" to be passed from node to node as MCS evolves, see [25].

Results: Random network. Figure 2 (bottom left) plots $\mathbf{err}_f$ versus the total number of transmissions. **AL–MG** and **AL–G** outperform PS. To decrease $\mathbf{err}_f$ below $5 \cdot 10^{-4}$, **AL–MG** and **AL–G** require about $1.2 \cdot 10^6$ transmissions, and **AL–G** $1.5 \cdot 10^6$ transmissions; PS requires about $3.7 \cdot 10^6$ transmissions to achieve the same precision. Figure 2 (bottom right) plots $\mathbf{err}_f$ plots versus the total number of floating point operations. PS requires less computation than **AL–G** and **AL–MG**. To decrease $\mathbf{err}_f$ below $5 \cdot 10^{-4}$, **AL–MG** and **AL–G** require about $69 \cdot 10^9$ transmissions; PS requires about $2.8 \cdot 10^9$ transmissions for same precision. With each of the algorithms **AL–G**, **AL–MG**, and PS, each node i's estimate x_i was feasible along time slots.

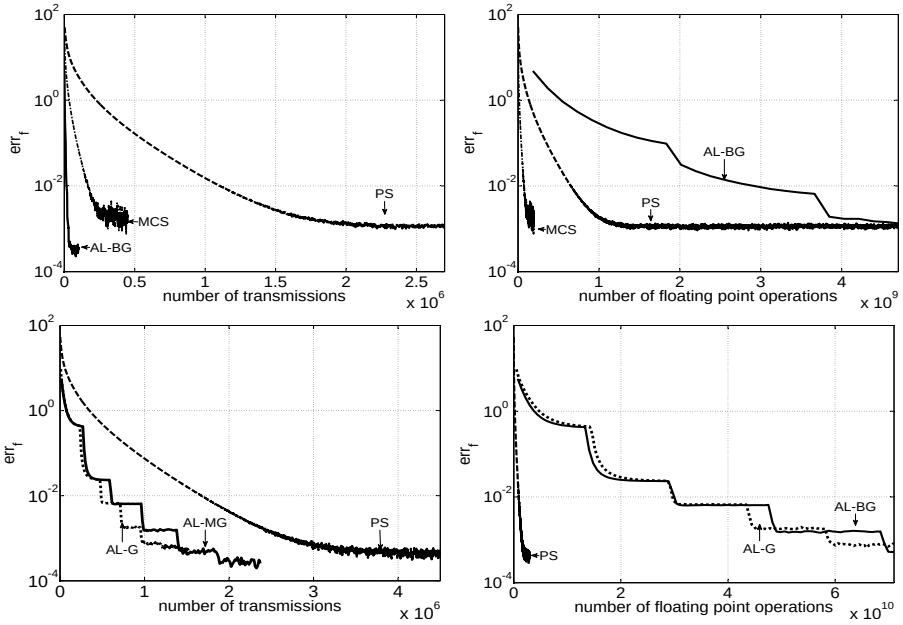

Fig. 2. Performance of **AL–BG**, MCS, and PS on a static network (top figures,) and the **AL–G**, **AL–MG** and PS algorithms on a random network (bottom figures.) Left: total number of transmissions; Right: total number of floating point operations.

B. Cooperative spectrum sensing for cognitive radio networks

We now consider cooperative spectrum sensing for cognitive radio networks. Cognitive radios are an emerging technology for improving the efficiency of usage of the radio spectrum. (For a tutorial on cognitive radios see, e.g., [29].) We focus here on the cooperative spectrum sensing approach that has been studied in [4], [3]. Suppose that N_r cognitive radios, located at x_r positions in 2D space, cooperate to determine: 1) the spatial locations; and 2) the power spectrum density (PSD) of primary users. Primary users can be located on N_s potential locations, x_s, on $\sqrt{N_s} \times \sqrt{N_s}$ square grid (See Figure 3, top, in [3].) For brevity, we omit the details of the problem setup; we refer to reference [4], subsection II-A, for the problem setup, and section II (eqn. (2)) in the same reference, for the Lasso optimization problem of estimating the locations and the PSD of primary users. This (unconstrained) optimization

problem in eqn. (2) in [4] fits the generic framework in eqn. (1); thus, our algorithms **AL–G**, **AL–MG** and **AL–BG** apply to solve the problem in eqn. (2) in [4]. Throughout, we use the same terminology and notation as in [4]. We now detail the simulation parameters. The number of potential sources is $N_s = 25$; they are distributed on a regular 5×5 grid over the square surface of 4km^2. Channel gains γ_{sr} are modeled as $\gamma_{sr} = \min\left\{1,\ \frac{A}{\|x_s - x_r\|^a}\right\}$, with $A = 200$ [meters] and a=3. The number of basis rectangles is $N_b = 6$, and the number of frequencies at which cognitive radios sample PSD is $N_f = 6$. There are 3 active sources; each source transmits at 2 out of $N_b = 6$ possible frequency bands. After some experimentation, we set the Lasso parameter λ (see eqn. (2) in [4]) to $\lambda = 1$; for a distributed algorithm to optimally set λ, see [4]. We consider the supergraph with $N_r = 20$ nodes (cognitive radios) and $|E| = 46$ undirected edges (92 arcs.) Nodes are uniformly distributed on a unit 2km×2km square and the pairs of nodes with distance smaller than r =750m are connected.

For static networks, we compare **AL–BG** (our algorithm) with MCS, PS, and an algorithm in [4]. Reference [4] proposes three (variants of AD-MoM type algorithms, mutually differing in: 1) the total number of primal and dual variables maintained by each node (cognitive radio); 2) the method by which nodes solve local optimizations for primal variable update (These problems are similar to (32).) We compare **AL–BG** with the DCD-Lasso variant, because it has the same number of primal and dual variables as **AL–BG** and a smaller computational cost than the alternative DQP-Lasso variant. With **AL–BG**, we use an increasing sequence of AL penalty parameters, $\rho_t = K_\rho\, A_\rho^t + C_\rho$, $t = 0, 1, ...$, with $K_\rho = 1$, $A_\rho = 1.15$ and $C_\rho = 3$. With DCD-Lasso, we used fixed $\rho = \rho_t$, as in [4], [3].[13] We solve the local problems in **AL–BG** (eqn. (32)), **AL–G** and **AL–MG** (eqn. (11),) by an efficient block coordinate method in [4] (see eqn. (13) in [4].) For the networks with link failures, we have compared our **AL–G** and **AL–MG** algorithms with PS (in [10], [11], [9], [8].) We briefly comment on the results. Both **AL–G** and **AL–MG** converge to a solution, in the presence of link failures as in VI-A; they converge slower than the PS algorithm, both in terms of communication and computational cost.

Results for static network. Figure 3 (left) plots $\mathbf{err}_f$ for PS, MCS, DCD-Lasso, and **AL–BG** versus the number of transmissions (at all nodes.) **AL–BG** shows improvement over the other algorithms. To achieve the precision of $\mathbf{err}_f \leq 0.044$, **AL–BG** requires about $5 \cdot 10^4$ transmissions; MCS $20 \cdot 10^4$ transmissions; DCD-Lasso $25 \cdot 10^4$ transmissions; PS $50 \cdot 10^4$ transmissions. Limiting error for PS is 0.027 (not visible in the plot.) Note also that DCD-Lasso and PS saturate at a larger error than **AL–BG** and MCS. Figure 3 (right) plots the $\mathbf{err}_f$ for the PS, MCS, DCD-Lasso, and **AL–BG** algorithms versus the total number of

[13]It may be possible to improve on the speed of DCD-Lasso by selecting appropriate time varying $\rho = \rho_t$; this is outside of our paper's scope.

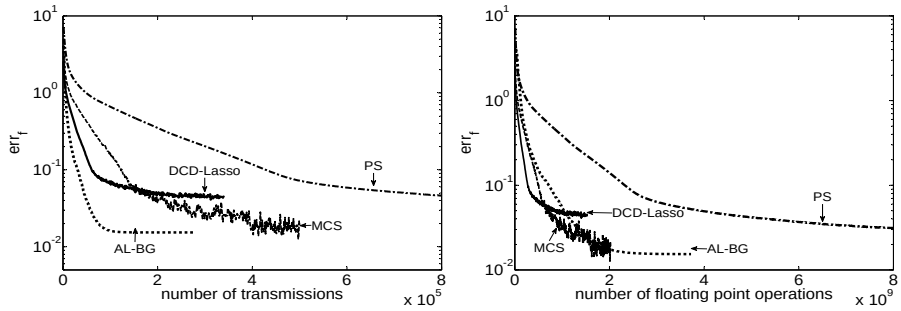

Fig. 3. Performance of **AL–BG**, DCD-Lasso, PS and MCS algorithms on static CR network. Left: total number of transmissions (cumulatively, at all nodes). Right: total number of floating point operations (cumulatively, at all nodes.)

floating point operations. **AL–BG**, MCS and DCD-Lasso show similar performance, while PS is slower.

VII. Conclusion

We studied cooperative optimization in networked systems, where each node obtains an optimal (scalar or vector) parameter of common interest, $x = x^\star$. Quantity $x^\star$ is a solution to the optimization problem where the objective is the sum of private convex objectives at each node, and each node has a private convex constraint on x. Nodes utilize gossip to communicate through a generic connected network with failing links. To solve this network problem, we proposed a novel distributed, decentralized algorithm, the **AL–G** algorithm. **AL–G** handles a very general optimization problem with private costs, private constraints, random networks, asymmetric link failures, and gossip communication.

This contrasts with existing augmented Lagrangian primal-dual methods that handle only static networks and synchronous communication, while, as mentioned, the **AL-G** algorithm handles random networks and uses gossip communication. In distinction with existing primal subgradient algorithms that essentially handle only symmetric link failures, **AL–G** handles asymmetric link failures.

AL–G updates the dual variables synchronously via a standard method of multipliers, and it updates the primal variables via a novel algorithm with gossip communication, the P–AL–G algorithm. P–AL–G is a nonlinear Gauss-Seidel type algorithm with random order of minimizations. Nonlinear Gauss-Seidel was previously shown to converge only under the *cyclic* or the *essentially cyclic* rules, [24], [6]; we prove convergence of P–AL–G, which has a *random* minimization order. Moreover, our proof is different from standard proofs for nonlinear Gauss-Seidel, as it uses as main argument the expected decrease in the objective function after one Gauss-Seidel step. We studied and proved convergence of two variants of **AL–G**, namely, **AL–MG** and **AL–BG**. An interesting future research direction is to develop a fully asynchronous primal-dual algorithm that updates both the dual and primal variables asynchronously.

The **AL–G** algorithm is a generic tool to solve a wide range of problems in networked systems; two simulation examples, l_1–regularized logistic regression for classification, and cooperative spectrum sensing for cognitive radios, demonstrated the applicability and effectiveness of **AL–G** in applications.

APPENDIX

Proof of Lemma 8. We first need a standard result from topology (proof omitted for brevity.)

Lemma 10 Let $\mathcal{X}$ and $\mathcal{Y}$ be topological spaces, where $\mathcal{Y}$ is compact. Suppose the function: $\kappa : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is continuous (with respect to the product topology on $\mathcal{X} \times \mathcal{Y}$ and the usual topology on $\mathbb{R}$; $\times$ denotes Cartesian product.) Then, the function $\gamma : \mathcal{X} \to \mathbb{R}$, $\gamma(a) := \inf\{\kappa(a, b) : b \in \mathcal{Y}\}$ is continuous.

Proof of Lemma 8: Denote by $\mathcal{P}_i : \mathbb{R}^{m(N+2M)} \to \mathbb{R}^m$ the projection map $\mathcal{P}_i(z) = z_i$, $i = 1, ..., N + 2M$. Further, denote by $\mathcal{P}_i(\Gamma(\epsilon)) :=\{z_i \in \mathbb{R}^m : \ z_i = \mathcal{P}_i(z), \text{ for some } z \in \Gamma(\epsilon)\}$. The set $\mathcal{P}_i(\Gamma(\epsilon))$ is compact, for all $i = 1, ..., N + 2M$, because the set $\Gamma(\epsilon)$ is compact. Consider now the set $\mathbb{R}^{m(N+2M)} \supset C_\epsilon := \mathcal{P}_1(\Gamma(\epsilon)) \times \mathcal{P}_2(\Gamma(\epsilon)) \times \ ... \ \times \mathcal{P}_{N+2M}(\Gamma(\epsilon))$, where the symbol $\times$ denotes the Cartesian product of the sets. Clearly, $C_\epsilon \supset \Gamma_\epsilon(B)$. We will show that L^i is continuous on C_ϵ, i.e., that $L^i : C_\epsilon \to \mathbb{R}$ is continuous, which will imply the claim of Lemma 8. Recall the definition of L^i in eqn. (18). It is easy to see that the minimum in eqn. (18) is attained on the set $\mathcal{P}_i\left(\Gamma(\epsilon)\right)$, i.e., that $L^i(z) = \min_{w_i\in\mathcal{P}_i(\Gamma(\epsilon))} L\left(z_1, z_2, ..., z_{i-1}, w_i, z_{i+1}, ..., z_{N+2M}\right)$. Thus, by Lemma 12, and because the function $L : \mathbb{R}^{m(N+2M)} \to \mathbb{R}$ is continuous, the function $L^i : \ \mathcal{P}_1(\Gamma(\epsilon)) \times \ ... \ \times \mathcal{P}_{i-1}(\Gamma(\epsilon)) \times \mathcal{P}_{i+1}(\Gamma(\epsilon)) \times ... \times \mathcal{P}_{N+2M}(\Gamma(\epsilon)) \to \mathbb{R}$ is continuous. But this means that $L^i : C_\epsilon \to \mathbb{R}$ is also continuous. ■

Convergence proof of the P–AL–MG algorithm. We first introduce an abstract model of the P–AL–MG algorithm. First, we impose an additional assumptions that the link failures are spatially independent, i.e., the Bernoulli states $A_{ij}(k)$ and $A_{lm}(k)$ of different links at time slot k are independent. Define the sets $Y(\Omega_i) := \{y_{ji} : \ j \in \Omega_i\}$ and the class $Y(O_i) := \{y_{ji} : \ j \in O_i\}$, where $O_i \subset \Omega_i$. One distinct set $Y(O_i)$ is assigned to each distinct subset O_i of Ω_i. (Clearly, $Y(\Omega_i)$ belongs to a class of sets $Y(O_i)$, as Ω_i is a subset of itself.) With P–AL–MG, at iteration k, minimization is performed either with respect to x_i, $i \in \{1, ..., N\}$, or with respect to some $Y(O_i)$. If none of the neighbors of node i receives successfully a message, then iteration k is void. Define the following collection of the subsets of primal variables: $\Pi := \{\{x_1\}, ..., \{x_N\}, Y(\Omega_1), ..., Y(\Omega_N)\}$. Collection Π constitutes a partition of the set of primal variables; that is, different subsets in Π are disjoint and their union contains all primal variables. Further, denote each of the subsets $\{x_i\}$, $Y(O_i)$, $Y(\Omega_i)$, with appropriately indexed Z_s, $s = 1, ..., S$. Then, with P–AL–MG, at time slot k, $L(z)$ is optimized with respect to one Z_s, $s = 1, ..., S$. Define $\xi(k)$, $k = 1, 2, ...$, as follows: $\xi(k) = s$, if, at time slot k, $L(z)$ is optimized with respect to Z_s; $\xi(k) = 0$, if, at k, no variable gets updated–when all transmissions at time slot k are unsuccessful. Denote by

$P(Z_s) = \text{Prob}\,(\xi(k) = s)$. Under spatial independence of link failures, $P(Z_s)$ can be shown to be strictly positive for all s. It can be shown that $\xi(k)$ are i.i.d. Consider now (16) and P–AL–MG. All results for P–AL–G remain valid for P–AL–MG also–only the expressions for the expected decrease of $L(\cdot)$ per iteration, $\psi(z)$, (Lemma 7), and the proof of Lemma 8 change. Denote by $L^{(Z_s)}(z)$ the optimal value after minimizing $L(\cdot)$ with respect to Z_s at point z (with the other blocks z_j, $z_j \notin Z_s$, fixed.) Then: $\psi(z) = \sum_{s=1}^{S} P(Z_s)\left(L^{(Z_s)} - L(z)\right)$. Recall $\phi(z) = -\psi(z)$ and the set $\Gamma(\epsilon)$, for some $\epsilon > 0$. Lemma 8 remains valid for P–AL–MG. To see this, first remark that $\phi(z) \geq 0$, for all $z \in F$. We want to show that $\phi(z) > 0$, for all $z \in \Gamma(\epsilon)$. Suppose not. Then, $L(z) = L^{(Z_s)}(z)$, for all Z_s, $s = 1, ..., S$. Then, in particular, $L(z) = L^{(Z_s)}(z)$, for all Z_s in the partition Π. Because $P(Z_s) > 0$, $\forall s$, this implies that the point z is block-optimal (where now, in view of Definition 2, Z_s is considered a single block). By Assumption 3, z is also optimal, which contradicts $z \in \Gamma(\epsilon)$. Thus, $\phi(z) > 0$ for all $z \in \Gamma(\epsilon)$. The proof now proceeds as with the proof of Lemma 8 for algorithm P–AL–G.

Convergence proof of the P–AL–BG algorithm. P–AL–BG is completely equivalent to P–AL–G, from the optimization point of view. P–AL–BG can be modeled in the same way as in Alg. 2, with a difference that, with P–AL–BG: 1) there is a smaller number ($= N$) of primal variables: $z_i := x_i$, $i = 1, ..., N$; and 2) $\text{Prob}(\zeta(k) = 0) = 0$. Thus, the analysis in section V is also valid for P–AL–BG.

REFERENCES

[1] L. Xiao, M. Johansson, and S. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *IEEE Transactions on Communications*, vol. 52, no. 7, pp. 1136–1144, January 2004.

[2] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," August 2008, submitted for publication, 30 pages. [Online]. Available: arXiv:0809.0009v1 [cs.MA]

[3] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1847–1862, March 2010.

[4] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, November 2010.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[6] D. Bersekas, *Nonlinear Programming*. Athena Scientific, 1995.

[7] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation*. New Jersey: Prentice-Hall, Englewood Cliffs, 1989.

[8] I. Lobel and A. Ozdaglar, "Convergence analysis of distributed subgradient methods over random networks," in *46th Annual Allerton Conference onCommunication, Control, and Computing*, Monticello, Illinois, September 2008, pp. 353 – 360.

[9] S. Ram, A. Nedic, and V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," submitted 2009. [Online]. Available: arXiv:0811.2595v1 [math.OC]

[10] I. Lobel, A. Ozdaglar, and D. Feijer, "Distributed multi-agent optimization with state-dependent communication," 2010, LIDS report 2834, Massachusetts Institute of Technology. Laboratory for Information and Decision Systems, Cambrige, MA, submitted for publication.

[11] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, January 2009.

[12] S. S. Ram, A. Nedic, and V. Veeravalli, "Asynchronous gossip algorithms for stochastic optimization," in *CDC '09, 48th IEEE International Conference on Decision and Control*, Shanghai, China, December 2009, pp. 3581 – 3586.

[13] S. S. Ram and A. Nedic, "A new class of distributed optimization algorithms: application to regression of distributed data," *Optimization Methods and Software*, 2010. [Online]. Available: http://www.informaworld.com/10.1080/10556788.2010.511669

[14] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," November 2010, unfinished working draft. [Online]. Available: http://www.stanford.edu/ boyd/papers/distr_opt_stat_learning_admm.html

[15] J.-B. H. Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms I: Fundamentals*. Springer Verlag, 1996.

[16] A. Nedic, "Optimization I," August 2008, lecture notes. [Online]. Available: https://netfiles.uiuc.edu/angelia/www/optimization_one.pdf

[17] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, July 2009.

[18] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315–3326, July 2008.

[19] T. Aysal, A. Sarwate, and A. Dimakis, "Reaching consensus in wireless networks with probabilistic broadcast," in *47th Annual Allerton Conference on Communication, Control and Computation*, Monticello, IL, Oct. 2009, pp. 732–739.

[20] R. T. Rockafellar, "A dual approach to solving nonlinear programming problems by unconstrained optimization," *Mathematical Programming*, vol. 5, no. 1, pp. 354–373, 1973.

[21] ——, "The multiplier method of Hestenes and Powell applied to convex programming," *Journal on Optimization Theory and Applications*, vol. 12, no. 6, pp. 133–145, 1973.

[22] D. Bertsekas, "Multiplier methods: a survey," *Automatica*, vol. 12, pp. 133–145, 1976.

[23] B. He and X. Yuan, "On the acceleration of augmented Lagrangian method for linearly constrained optimization." [Online]. Available: http://www.optimization-online.org/DB_HTML/2010/10/2760.html

[24] P. Tseng, "Convergence of block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, June 2001.

[25] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1157–1170, August 2009.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[28] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN '05, Information Processing in Sensor Networks*, Los Angeles, California, 2005, pp. 63–70.

[29] X. Hong, C. Wang, H. Chen, and Y. Zhang, "Secondary spectrum access networks," *IEEE Veh. Technol. Mag.*, vol. 4, no. 2, pp. 36–43, 2009.